

Przegląd metod i algorytmów numerycznych

Część 1

Janina i Michał Jankowscy +

Wydanie drugie poprawione



Wydawnictwa Naukowo-Techniczne
Warszawa 1988

JANINA JANKOWSKA
Instytut Informatyki UW

MICHAŁ JANKOWSKI
Instytut Informatyki UW

© Copyright by Janina Jankowska
and Michał Jankowski
Warszawa 1981

© Copyright by Wydawnictwa
Naukowo-Techniczne
Warszawa 1988

All rights reserved
Printed in Poland

ISBN 83-204-1039-8 całość
83-204-1040-1 część 1

Przegląd metod i algorytmów numerycznych. Część 1

Survey of numerical methods
and algorithms. Part 1

English summary, see p. 213

Обзор численных методов
и алгоритмов. Часть 1

Резюме, см. стр. 214

422580 / 02.1.

Książka stanowi pierwszą część monografii poświęconej przeglądowi metod numerycznych. Omówiono w niej własności arytmetyki zmiennopozycyjnej oraz podstawowe metody i algorytmy rozwiązywania zadań interpolacyjnych i aproksymacyjnych, przybliżonego obliczania całek oraz rozwiązywania równań różniczkowych zwyczajnych. Specjalną uwagę poświęcono ocenie numerycznej stabilności algorytmów i porównywaniu ich efektywności ze złożonością obliczeniową odpowiednich zadań. Książka jest przeznaczona dla programistów, pracowników nauki zajmujących się informatyką oraz dla studentów kierunków informatycznych.

518:681.3

1000079245



Redaktor pierwszego wydania

BARBARA KALISZ

Okładkę i układ typograficzny serii
projektował

TADEUSZ PIETRZYK

Redaktor techniczny

BARBARA CHOJNACKA-FLISIUK

Tytuł dotowany przez
Ministra Edukacji Narodowej

UW
Lublin

EO. 120/89/16

3 02. 89r.

Spis treści

Przedmowa		7
1	Podstawowe pojęcia związane z obliczeniami numerycznymi	9
1.1	Numeryczna reprezentacja liczb	9
1.2	Uwarunkowanie zadania	11
1.3	Zmiennopozycyjne działania arytmetyczne	14
1.4	Algorytmy numerycznie poprawne	17
1.5	Algorytmy numerycznie stabilne	23
1.6	Arytmetyki wyższej precyzji	29
1.7	Informacja o obliczaniu funkcji standardowych	33
1.8	Złożoność obliczeniowa	34
1.9	Uwagi	36
2	Interpolacja	37
2.1	Postawienie zadania interpolacyjnego	37
2.2	Postacie wielomianu, obliczanie wartości wielomianu i jego pochodnych	38
2.3	Interpolacja Lagrange'a	45
2.3.1	Sformułowanie zadania	45
2.3.2	Ilorazy różnicowe (przypadek węzłów jednokrotnych)	47
2.3.3	Interpolacja Lagrange'a w przypadku węzłów równoodległych	53
2.4	Interpolacja Hermite'a	55
2.4.1	Sformułowanie zadania	55
2.4.2	Ilorazy różnicowe	56
2.5	Reszta wzoru interpolacyjnego, problem zbieżności procesu interpolacji	60
2.6	Interpolacja trygonometryczna	64
2.6.1	Sformułowanie zadania	64
2.6.2	Szybkie transformacje Fouriera (FFT)	67
2.6.3	Algorytmy Goertzel'a i Reinscha	73
2.7	Interpolacja funkcjami sklejanymi	75
3	Aproksymacja	85
3.1	Zadanie aproksymacji funkcji w przestrzeniach unormowanych	85
3.2	Przestrzeń unitarna. Algorytmy ortogonalizacyjne	89

3.3	Ciągi wielomianów ortogonalnych	92
3.3.1	Definicje i ogólne własności	92
3.3.2	Wielomiany Czebyszewa	98
3.4	Aproksymacja w przestrzeniach unitarnych	101
3.4.1	Przypadek ogólny	101
3.4.2	Aproksymacja średniokwadratowa wielomianami	107
3.5	Aproksymacja jednostajna	110
3.6	Informacja o innych rodzajach aproksymacji	122

4 Kwadratury 127

4.1	Wstęp	127
4.2	Liniowa zamiana zmiennej w całce i w kwadraturze	129
4.3	Problem zbieżności ciągu kwadratur	131
4.4	Kwadratury interpolacyjne	133
4.5	Kwadratury Newtona-Cotesa	136
4.6	Kwadratury Gaussa	145
4.7	Kwadratury o minimalnej reszcie	150
4.8	Przyspieszanie szybkości zbieżności ciągu kwadratur	153
4.9	Obliczanie całek niewłaściwych i całek funkcji z osobliwościami	156
4.10	Uwagi o wyborze kwadratur i ich numerycznym obliczaniu	161
4.11	Informacje o metodach obliczania całek wielokrotnych	164

5 Równania różniczkowe zwyczajne 167

5.1	Zagadnienie początkowe Cauchy'ego	167
5.2	Metody różnicowe (metody dyskretyzacji)	169
5.3	Metoda Eulera i jej modyfikacje	170
5.4	Metody jednokrokowe	173
5.4.1	Aproksymacja i zbieżność	173
5.4.2	Rząd i błąd metody jednokrokowej	177
5.4.3	Wpływ błędów zaokrągleń w arytmetyce zmiennopozycyjnej. Wybór optymalnego kroku całkowania	180
5.4.4	Metody Rungego-Kutty	182
5.4.5	Asymptotyczna postać błędu w metodach jednokrokowych	186
5.5	Liniowe metody wielokrokowe	187
5.5.1	Wprowadzenie	187
5.5.2	Przykłady konstrukcji liniowych metod wielokrokowych	189
5.5.3	Stabilność i aproksymacja metod wielokrokowych	190
5.5.4	Problem zbieżności, oszacowanie błędu metody i błędu wytworzonego w arytmetyce zmiennopozycyjnej	194
5.5.5	Przykłady ilustrujące różne rodzaje stabilności	197
5.5.6	Metody otwarte i zamknięte. Metody typu predyktor-korektor	200
5.5.7	Uwagi o rozwiązywaniu zagadnień początkowych dla układów równań zwyczajnych pierwszego rzędu	203
5.6	Uwagi o rozwiązywaniu układów sztywnych	203

Literatura 206

Skorowidz 209

Przedmowa

Książka niniejsza stanowi pierwszą część monografii poświęconej najważniejszym metodom i algorytmom numerycznym. Ze względu na jej niewielką objętość musieliśmy ograniczyć się do rozpatrzenia tylko wybranych zadań. Staraliśmy się, aby były to problemy najczęściej występujące w praktyce obliczeniowej.

W pierwszym rozdziale omawiamy własności arytmetyki numerycznej i związane z nią zagadnienia. Następne rozdziały poświęciliśmy kolejno: interpolacji, aproksymacji, kwadraturom (tj. przybliżonemu obliczaniu całek) i rozwiązywaniu równań różniczkowych zwyczajnych. Natomiast w drugiej części monografii Czytelnik znajdzie omówienie metod numerycznych rozwiązywania układów równań liniowych oraz równań i układów równań nieliniowych, wyznaczania wartości i wektorów własnych macierzy, rozwiązywania liniowego zagadnienia najmniejszych kwadratów i wreszcie metod przybliżonego rozwiązywania zagadnień z równaniami różniczkowymi cząstkowymi.

Spośród, na ogół wielu, znanych metod rozwiązywania ustalonego zadania przedstawiamy metody mało kosztowne, gwarantujące zarazem możliwie dużą dokładność wyniku. Tam gdzie potrafimy, porównujemy koszt metody ze złożonością obliczeniową zadania, tzn. z minimalną liczbą działań arytmetycznych potrzebnych dla jego rozwiązania. Duży nacisk kładziemy także na badanie numerycznej stabilności algorytmów, przez którą rozumiemy ich „odporność” na błędy wytworzone w działaniach realizowanych w arytmetyce zmiennopozycyjnej.

Uważamy, że w praktyce tylko w przypadku bardzo niewielu zadań rozwiązywanych numerycznie wystarczy sięgnąć po gotową receptę na rozwiązanie. Na ogół jest potrzebna znacznie głębsza analiza problemu i odpowiednia interpretacja otrzymanych wyników. Dlatego też naszym celem było nie tyle podanie „przepisów” rozwiązywania zadań, co raczej pokazanie różnych możliwości i dróg prowadzących do rozwiązań. Konsekwencją tego jest przytaczanie sporej liczby twierdzeń. Sądzymy, że przedstawione dowody wybranych twierdzeń pozwalają na lepsze zrozumienie własności metod i często wyjaśniają konstrukcję algorytmów. Staraliśmy się sygnalizować nowe wyniki, a także te metody, których ze względu na

brak miejsca nie mogliśmy dokładnie przedstawić. Mamy nadzieję, że zainteresowani sięgną po odpowiednie pozycje literatury.

Czytelnik nie znajdzie w tej książce ani jednej procedury. Jedynie fragmenty niektórych algorytmów zapisujemy w Algolu (a i to najczęściej nie w języku wzorcowym). Podajemy natomiast, tam gdzie to jest możliwe, dane bibliograficzne o dostępnych, dobrze przetestowanych i opisanych procedurach w Algolu 60 i Fortranie. Niezłym źródłem wydaje się też być tworzona obecnie przez Elwro „Biblioteka modułów matematycznych maszyn cyfrowych Jednolitego Systemu” zawierająca już około 700 procedur w Fortranie.

Na zakończenie chcielibyśmy podziękować Opiniodawcy i tym wszystkim, których krytyczne uwagi przyczyniły się do usunięcia licznych usterek i wprowadzenia zmian w tej książce. Szczególną wdzięczność za okazaną pomoc i rady jesteśmy winni Andrzejowi Kielbasińskiemu i Henrykowi Woźniakowskiemu. Fakt, że mimo wszystko, książka nie jest wolna od błędów, jest naszą winą.

J. i M. JANKOWSCY

Warszawa, w grudniu 1978

Numeryczna reprezentacja liczb

1.1

Liczby są reprezentowane w maszynie cyfrowej przez skończoną liczbę cyfr ich rozwinięć pozycyjnych. Najczęściej stosowanymi podstawami tych rozwinięć są 2, 8 lub 16. Bez zmniejszania ogólności ograniczymy się dalej do pierwszej z nich, tzn. będziemy rozpatrywali *arytmetykę dwójkową* (binarną). Nie chcemy zajmować się szczegółami jej różnych praktycznych realizacji (np. arytmetyka uzupełnieniowa, skalowanie liczb itp.) odsyłając zainteresowanych do podręczników [28] i [69]. Własności arytmetyk większości maszyn cyfrowych pracujących w Polsce są takie same jak własności przedstawianej niżej arytmetyki modelowej.

Sposób reprezentacji liczby zależy od jej typu, liczby całkowite (typu **integer**) są przedstawiane w sposób *stałopozycyjny*, a rzeczywiste (typu **real**) w sposób *zmiennopozycyjny*. Zaczniemy od reprezentacji stałopozycyjnej. Dowolną liczbę całkowitą l możemy przedstawić w postaci rozwinięcia dwójkowego

$$l = s \sum_{i=0}^n e_i 2^i, \quad (e_i \neq 0 \quad \text{dla} \quad i \leq n) \quad (1.1)$$

gdzie s jest znakiem liczby l ($s = +1$ lub -1), a $e_i = 0$ lub 1 są jej cyframi dwójkowymi. W maszynie cyfrowej na reprezentację liczby przeznaczona jest skończona długość np. $d+1$ bitów (cyfr dwójkowych). Jeśli tylko $n < d$, to liczba l jest reprezentowalna w rozpatrywanej arytmetyce. Często jest przedstawiana przez słowo postaci

s	0	$\dots$	0	e_n	e_{n-1}	$\dots$	e_0
-----	-----	---------	-----	-------	-----------	---------	-------

↑
znak

⏟
 d cyfr dwójkowych

(1.2)

W ten sposób mogą być reprezentowane *dokładnie* liczby całkowite l z przedziału $[-2^d + 1, 2^d - 1]$. Dla współczesnych maszyn cyfrowych $d \in (20, 60)$, co oznacza, że w przybliżeniu $2^d \in (10^6, 10^{18})$. Przy założeniu, że argumenty i wynik są reprezentowalne, dodawanie, odejmowanie i mnożenie liczb całkowitych (typu **integer**) jest wykonywane *dokładnie*. Zobaczmy, że na ogół nie jest tak dla liczb rzeczywistych (typu **real**).

Dowolną liczbę rzeczywistą $x \neq 0$ możemy przedstawić w postaci

$$x = s \cdot 2^c m \quad (1.3)$$

gdzie $s = +1$ lub -1 jest jej znakiem,

c jest liczbą całkowitą zwaną *cechą*,

m jest liczbą rzeczywistą z przedziału $[\frac{1}{2}, 1)$ nazywaną *mantysą*.

Dla $x \neq 0$ przedstawienie (1.3) jest jednoznaczne. Cechę c zapisuje się w sposób stałopozycyjny (1.2) na $d-t$ bitach słowa maszyny cyfrowej. Zakładamy tu, że c należy do przedziału liczb reprezentowalnych w ten sposób. Pozostałych t bitów słowa przeznacza się na reprezentację mantysy m . A więc zamiast na ogół nieskończonego rozwinięcia mantysy

$$m = \sum_{i=1}^{\infty} e_{-i} \cdot 2^{-i}, \quad (e_{-1} = 1; \quad e_{-i} = 0 \text{ lub } 1 \text{ dla } i > 1)$$

zapamiętywanych jest tylko jej t początkowych cyfr dwójkowych. Dokładniej, zakładamy, że mantysa została prawidłowo zaokrąglona do t cyfr, tzn.

$$m_t = \sum_{i=1}^t e_{-i} \cdot 2^{-i} + e_{-(t+1)} \cdot 2^{-t}$$

a wtedy

$$|m - m_t| \leq \frac{1}{2} \cdot 2^{-t} \quad (1.4)$$

Liczba x jest zatem reprezentowana trójką (s, c, m_t) zapamiętywaną np. w słowie postaci

s	s_c	e_{d-t-1}	e_{d-t-2}	$\dots$	e_0	e_{-1}	e_{-2}	$\dots$	e_{-t}
$d-t$ bitów cechy						t bitów mantysy			

Zero jest na ogół reprezentowane słowem o wszystkich bitach zerowych. Reprezentację zmiennopozycyjną liczby x będziemy oznaczali przez $rd(x)$

$$rd(x) = s \cdot 2^c m_t$$

Ze wzoru (1.4) wynika, że dla $x \neq 0$

$$\left| \frac{rd(x) - x}{x} \right| \leq 2^{-t}$$

co jest równoważne relacji

$$rd(x) = x(1 + \varepsilon), \quad \text{gdzie } |\varepsilon| \leq 2^{-t} \quad (1.5)$$

Z dotychczasowych rozważań widzimy, że liczby rzeczywiste są na ogół reprezentowane nie dokładnie, ale w sposób przybliżony z błędem względnym ε nie większym od 2^{-t} . Dla współczesnych maszyn cyfrowych $2^{-t} \in (10^{-15}, 10^{-6})$. Konsekwencjami przybliżonego reprezentowania danych zajmiemy się w dalszej części tego rozdziału.

Zauważmy, że liczba cyfr mantysy decyduje o dokładności zmiennopozycyjnego przedstawienia liczb, a liczba cyfr cechy określa zakres reprezentowalnych liczb. Przy przyjętych przez nas założeniach cecha $c \in [c_{\min}, c_{\max}]$, gdzie $c_{\max} = -c_{\min} = 2^{d-t} - 1$, a stąd i z założenia, że $m_t \in [\frac{1}{2}, 1)$ wynika, że w arytmetyce zmiennopozycyjnej możemy reprezentować liczby $x \neq 0$ spełniające zależności

$$\frac{1}{2} 2^{c_{\min}} \leq |x| < 2^{c_{\max}} \quad (1.6)$$

Liczy, których cecha c jest mniejsza niż $c_{\min}$, są reprezentowane zerem. Sytuację taką nazywamy *niedomiarem*. Nie jest wtedy spełniona relacja (1.5), gdyż błąd względny takiej reprezentacji jest równy 100%. Może to powodować nagłą poważną utratę dokładności obliczeń lub niekiedy ich alarmowe przerwanie.

W przypadku pojawienia się liczby, której cecha jest większa od $c_{\max}$, tzn. *nadmiaru*, zawsze następuje przerwanie obliczeń.

We współczesnych maszynach cyfrowych zakres reprezentowalnych liczb rzeczywistych jest tak duży (przykładowo dla maszyn IBM 360/370 i R-22,32 jest $5.4_{10} - 79 \lesssim |x| \lesssim 7.2_{10} 75$, a dla CDC 6400 jest $10 - 293 \lesssim |x| \lesssim 10 322$), że problem niedomiaru lub nadmiaru pojawia się rzadko w praktyce obliczeniowej. Dla uniknięcia go wystarcza niekiedy prosta modyfikacja algorytmu.

PRZYKŁAD 1.1

Obliczamy $|z|$, gdzie z jest liczbą zespoloną

$$z = a + bi \neq 0, \quad i = \sqrt{-1}$$

W klasycznym algorytmie

$$|z| := \text{sqrt}(a \times a + b \times b)$$

wystąpi nadmiar (lub niedomiar), jeśli a lub b są co do modułu większe od $2^{c_{\min}/2}$ (mniejsze np. od $\frac{1}{2} \cdot 2^{c_{\min}/2}$). Przekształcenie algorytmu (zapisujemy go jak poprzednio w pseudo-Algolu)

$$|z| := \text{if } |a| > |b| \text{ then } |a| \times \text{sqrt}(1 + (|b|/|a|) \uparrow 2) \\ \text{else } |b| \times \text{sqrt}(1 + (|a|/|b|) \uparrow 2)$$

pozwala na poprawne obliczanie $|z|$ dla dowolnych a i b spełniających (1.6). $\square$

Uwarunkowanie zadania

1.2

Zajmijmy się teraz konsekwencjami wynikającymi z przybliżonej reprezentacji liczb. Ograniczamy się do zadań polegających na obliczeniu dla danych $\mathbf{d} = (d_1, d_2, \dots, d_n)$ wyniku $\mathbf{w} = (w_1, w_2, \dots, w_m)$

$$\mathbf{w} = \varphi(\mathbf{d})$$

gdzie φ jest ciągłym odwzorowaniem, $\varphi: R_d \rightarrow R_w$, określonym na podzbiorze $D_0 \subset R_d$ o wartościach w R_w , a R_d i R_w są skończone wymiarowymi unormowanymi przestrzeniami kartezjańskimi.

Rozwiązując numerycznie takie zadanie musimy zdawać sobie sprawę z tego, że zamiast dokładnymi danymi $d_1, d_2, \dots, d_n$ ($d_i \in R$), dysponujemy tylko ich reprezentacjami $rd(d_1), rd(d_2), \dots, rd(d_n)$. Zgodnie ze wzorem (1.5) $rd(d_i) = d_i(1 + \varepsilon_i)$ i $|\varepsilon_i| \leq 2^{-i}$ dla $i = 1, 2, \dots, n$. Ta niewielka względna zmiana danych może jednak powodować duże zmiany względne rozwiązania.

PRZYKŁAD 1.2 (J. H. Wilkinson [80, s. 62])

Zerami wielomianu

$$w(x) = a_{20}x^{20} + \dots + a_1x + a_0 = \prod_{k=1}^{20} (x - k)$$

są liczby naturalne $1, 2, \dots, 20$. Zaburzając jeden z danych współczynników, np. $a_{19} = -210$, otrzymujemy $w_\varepsilon(x) = w(x) - 2^{-23}x^{19}$ o współczynniku $a_{19}(\varepsilon) = -(210 + 2^{-23})$. A zatem $a_{19}(\varepsilon) = a_{19}(1 - \varepsilon)$, gdzie $\varepsilon = (2^{23} \cdot 210)^{-1} < 2^{-30}$. Okazuje się jednak, że ta niewielka względna zmiana tylko jednego współczynnika wielomianu powoduje duże zmiany jego zer. Wielomian $w_\varepsilon(x)$ ma już pierwiastki zespolone i np. najbliższymi pierwiastkami 15 wielomianu $w(x)$ są pierwiastki $13.992358137 \pm 2.518830070i$. $\square$

Jeśli niewielkie względne zmiany danych zadania powodują duże względne zmiany jego rozwiązania, to zadanie takie nazywamy *źle uwarunkowanym*. Wielkości charakteryzujące wpływ zaburzeń danych na zaburzenia rozwiązania nazywamy *wskaźnikami uwarunkowania* zadania. Dla pewnej klasy zadań zdefiniujemy je ściśle w p. 1.5.

Badaniem uwarunkowania zer wielomianu $w(x)$ z przykładu 1.2 zajmiemy się dokładniej w przykładzie 7.9 części drugiej. W tym miejscu ograniczymy się do wyjaśnienia wprowadzonych pojęć na dwóch nieco prostszych przykładach.

PRZYKŁAD 1.3

Zbadajmy uwarunkowanie zadania obliczenia iloczynu skalarnego

$$S = \sum_{i=1}^n a_i b_i \neq 0$$

W tym celu zaburzmy dane $\mathbf{a} = (a_1, a_2, \dots, a_n)$ i $\mathbf{b} = (b_1, b_2, \dots, b_n)$ mnożąc ich współrzędne odpowiednio przez $1 + \alpha_i$ i $1 + \beta_i$ ($i = 1, 2, \dots, n$). Zakładamy, że α_i i β_i są na tyle małe, aby ich iloczyny można było zaniedbać. Szacując względną zmianę wyniku otrzymujemy

$$\begin{aligned} & \left| \frac{\sum_{i=1}^n a_i(1 + \alpha_i) b_i(1 + \beta_i) - \sum_{i=1}^n a_i b_i}{\sum_{i=1}^n a_i b_i} \right| \approx \left| \frac{\sum_{i=1}^n a_i b_i(\alpha_i + \beta_i)}{\sum_{i=1}^n a_i b_i} \right| \leq \\ & \leq \max_i |\alpha_i + \beta_i| \left(\sum_{i=1}^n |a_i b_i| / \left| \sum_{i=1}^n a_i b_i \right| \right) \end{aligned}$$

A więc za wskaźnik uwarunkowania możemy uznać wielkość $\text{cond}(a, b) = \sum_{i=1}^n |a_i b_i| / |\sum_{i=1}^n a_i b_i|$, gdyż maksymalne zaburzenie względne danych może się przenieść na zaburzenie względne wyniku co najwyżej z takim mnożnikiem. Oszacowanie to jest osiągane przy odpowiednich wartościach a_i, b_i, α_i oraz β_i . W przypadku, w którym wszystkie sumowane składniki $a_i b_i$ są tego samego znaku, $\text{cond}(a, b) = 1$, a więc zadanie obliczenia iloczynu skalarnego S jest bardzo dobrze uwarunkowane (mało wrażliwe na zaburzenia danych). $\square$

PRZYKŁAD 1.4

Rozważmy zadanie wyznaczenia rzeczywistych zer x_1, x_2 trójkątniku kwadratowego $x^2 - 2px + q$ dla danych p, q ze zbioru $D = \{(p, q): p \neq 0, q \neq 0, p^2 - q > 0\}$. Pierwiastki $x_1 = x_1(p, q), x_2 = x_2(p, q)$ są w zbiorze D ciągłymi i różniczkowalnymi funkcjami p i q . W ustalonym punkcie $(p_0, q_0) \in D$ mamy zatem dla $k = 1, 2$

$$\begin{aligned} \frac{x_k(p_0(1+\delta p_0), q_0(1+\delta q_0)) - x_k(p_0, q_0)}{x_k} &\approx \\ &\approx \frac{\partial x_k}{\partial p} \bigg|_{(p_0, q_0)} \frac{p_0}{x_k} \delta p_0 + \frac{\partial x_k}{\partial q} \bigg|_{(p_0, q_0)} \frac{q_0}{x_k} \delta q_0 \end{aligned}$$

gdzie δp_0 i δq_0 są małymi względnymi zaburzeniami p_0 i q_0 . Korzystając z reguł różniczkowania funkcji uwikłanych łatwo otrzymujemy zależność wskaźników uwarunkowania cond_p^k i cond_q^k od p_0 i q_0

$$\text{cond}_p^k(p_0, q_0) \stackrel{\text{dr}}{=} \frac{\partial x_k}{\partial p} \bigg|_{(p_0, q_0)} \frac{p_0}{x_k} = \frac{x_k}{x_k - p_0} \frac{p_0}{x_k}$$

A stąd, korzystając ze wzorów na pierwiastki trójkątniku kwadratowego

$$x_1 = p + \sqrt{p^2 - q}, \quad x_2 = p - \sqrt{p^2 - q}$$

dostajemy

$$\text{cond}_p^1(p_0, q_0) = -\text{cond}_p^2(p_0, q_0) = \frac{1}{\sqrt{1 - q_0/p_0^2}}$$

Analogicznie

$$\text{cond}_q^k(p_0, q_0) \stackrel{\text{dr}}{=} \frac{\partial x_k}{\partial q} \bigg|_{(p_0, q_0)} \frac{q_0}{x_k} = \frac{-q_0}{2x_k(x_k - p_0)}$$

i po przekształceniach otrzymujemy

$$\text{cond}_q^1(p_0, q_0) = \frac{1 - \sqrt{1 - q_0/p_0^2}}{2\sqrt{1 - q_0/p_0^2}} \quad \text{oraz}$$

$$\text{cond}_q^2(p_0, q_0) = \frac{1 + \sqrt{1 - q_0/p_0^2}}{-2\sqrt{1 - q_0/p_0^2}}$$

Z zależności tych wynika, że dla $q/p^2 \approx 1$ zadanie jest bardzo źle uwarunkowane (dla $k = 1, 2$ $\text{cond}_p^k, \text{cond}_q^k \rightarrow \pm \infty$, przy $q/p^2 \rightarrow 1$), natomiast dla $q/p^2 \ll 1$ jest bardzo dobrze uwarunkowane. $\square$

Podkreślmy jeszcze raz, że uwarunkowanie jest cechą samego zadania, niczym nie związaną z metodą jego rozwiązywania. Często natomiast możemy od danego problemu technicznego czy fizycznego przejść do różnych, matematycznie równoważnych zadań, które będziemy rozwiązywać numerycznie. Jest istotne, aby z tych możliwości wybierać tę, która prowadzi do zadania możliwie najlepiej uwarunkowanego (abstrahujemy tu od kosztu rozwiązywania). Niejednokrotnie warto a priori rozstrzygnąć, czy rozwiązywanie danego zadania ma w ogóle sens w konkretnej arytmetyce. Zdarza się bowiem, że zadanie jest tak bardzo wrażliwe na zaburzenia danych (tak źle uwarunkowane), że błąd ich reprezentacji może spowodować zaburzenie względne rozwiązania nawet na poziomie jedności. Rozwiązywanie zadania w tej sytuacji jest oczywiście niecelowe.

PRZYKŁAD 1.5

Rozpatrzmy zadanie rozwiązania układu równań liniowych $Ax = b$ o nieosobliwej macierzy A . W arytmetyce zmiennopozycyjnej reprezentacją macierzy A jest $rd(A) = A + E$ ($E = (e_{ij})$, $|e_{ij}| \leq 2^{-t}|a_{ij}|$), a reprezentacją wektora b jest $rd(b) = b + e$ ($|e_i| \leq 2^{-t}|b_i|$). Jeżeli $\|A\| \|A^{-1}\| \geq 2^t$, to macierz $A + E$ może być osobliwa. Ponadto, jak pokażemy na str. 22 w rozdz. 6 części drugiej, zaburzenie względne $\|e\|/\|b\|$ samego wektora prawej strony b , które dla większości norm jest rzędu 2^{-t} , może przenieść się na zaburzenie względne rozwiązania z mnożnikiem $\|A\| \|A^{-1}\|$. Wielkość ta (wskaźnik uwarunkowania zadania) może być dowolnie duża. Widzimy stąd, że błąd reprezentacji danych istotnie ogranicza klasę zadań, które jest sens rozwiązywać w danej arytmetyce. Dla układów liniowych są to zadania o macierzy A , dla której wskaźnik uwarunkowania $\|A\| \|A^{-1}\|$ jest mniejszy od 2^t . $\square$

Zmiennopozycyjne działania arytmetyczne

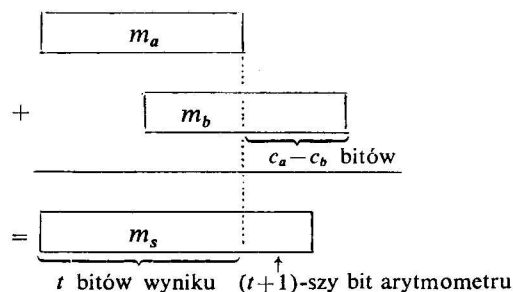
1.3

Mówiąc o działaniach arytmetycznych zakładamy zawsze, że ich argumenty są równe swoim reprezentacjom zmiennopozycyjnym oraz, że przy obliczaniu wyniku nie wystąpi nadmiar ani niedomiar. Realizację działań w arytmetyce zmiennopozycyjnej, oznaczonej dalej skrótem *fl* (od angielskiego terminu floating point), omówimy na przykładzie dodawania. Niech $a = rd(a) = 2^{c_a}m_a$ i $b = rd(b) = 2^{c_b}m_b$ będą danymi argumentami, dla uproszczenia takimi, że $a \geq b > 0$. Ich dokładna suma jest równa

$$a + b = 2^{c_a}(m_a + m_b 2^{-(c_a - c_b)}) = 2^{c_a}m_s \quad (1.7)$$

* Definicje i własności norm wektorów i macierzy znajdzie Czytelnik w rozdz. 6 części drugiej.

Mantysa wyniku będzie obliczana przez dodanie do mantysy liczby a mantysy liczby b pomnożonej przez $2^{-(c_a - c_b)}$. Mnożenie liczby przez całkowitą potęgę podstawy danego systemu pozycyjnego (tutaj równej 2) odpowiada „przesunięciu” rozwinięcia tej liczby w zapisie pozycyjnym o tyle miejsc, ile wynosi wartość wykładnika potęgi. Jeśli wykładnik jest ujemny, to „przesuwamy” liczbę w prawo, a gdy dodatni, to w lewo. Rozważany przez nas przypadek obliczania mantysy wyniku możemy zatem zilustrować następująco:



Zakładamy, że mantysa m_s wyniku zostanie obliczona w arytmometrze o co najmniej $t+1$ bitach (w wielu maszynach cyfrowych działania arytmetyczne są wykonywane na rejestrach podwójnej długości, tzn. na $2t$ bitach), a więc będzie mogła być prawidłowo zaokrąglona do t początkowych cyfr dwójkowych. Nastąpi to po jej ewentualnym znormalizowaniu (podzieleniu przez 2), gdy $m_s \geq 1$. Zawsze jednak będzie spełniony warunek (1.4), a stąd dla obliczonej w arytmetyce fl sumy a i b wynika zależność

$$fl(a+b) = rd(a+b)$$

Ogólnie

$$fl(a \square b) = rd(a \square b) = (a \square b)(1 + \varepsilon), \quad \varepsilon = \varepsilon(a, b, \square), \quad |\varepsilon| \leq 2^{-t} \quad (1.8)$$

gdzie $fl(a \square b)$ oznacza wynik zmiennopozycyjnego działania $\square$ ($\square = +, -, *, /$) na argumentach a i b równych swoim reprezentacjom $a = rd(a)$, $b = rd(b)$.

Dla uproszczania oszacowań błędów wytworzonych wprowadzamy (por. pracę [42]) relację równości przybliżonej $\stackrel{1}{=}$.

DEFINICJA 1.1

Jeśli $a(t)$ i $b(t)$ są funkcjami zmiennej t , $t \geq 1$, to $a(t) \stackrel{1}{=} b(t)$ oznacza, że istnieją stałe $K \geq 0$ i $t_0 \geq 1$ takie, że

$$|a(t) - b(t)| \leq K \cdot 2^{-t} |b(t)| \quad \text{dla } t > t_0$$

Nierówność przybliżona $a(t) \leq b(t)$ oznacza $a(t) \leq b(t)$ lub $a(t) \stackrel{1}{=} b(t)$. ■

Przykłady:

$$1 + 2n \cdot 2^{-t} = 1;$$

jeśli np. $|\varepsilon| \leq 2L \cdot 2^{-t}$, to $\sqrt[1]{1+\varepsilon} = 1 + \delta$, gdzie $|\delta| \leq L \cdot 2^{-t}$;

jeśli $|\varepsilon_i| \leq 2^{-t}$ dla $i = 1, 2, \dots, n$, to $\prod_{i=1}^n (1 + \varepsilon_i) = 1 + E$, gdzie $|E| \leq n \cdot 2^{-t}$.

Pozornie mogłoby się wydawać, że niewielkie ($2^{-t} \in [10^{-15}, 10^{-6}]$) względne błędy wytworzone w poszczególnych działaniach arytmetycznych nie powinny przysparzać istotnych kłopotów w obliczeniach numerycznych. Nie jest to jednak prawdą nawet dla bardzo prostych zadań.

PRZYKŁAD 1.6

Rozpatrzmy dwa różne algorytmy obliczania różnicy kwadratów dwóch liczb:

$$A1(a, b) = a^2 - b^2$$

$$A2(a, b) = (a - b)(a + b)$$

Przy realizacji pierwszego z nich w arytmetyce fl otrzymujemy

$$\begin{aligned} fl(a^2 - b^2) &= (a \times a(1 + \varepsilon_1) - b \times b(1 + \varepsilon_2))(1 + \varepsilon_3) \\ &= (a^2 - b^2) \left(1 + \frac{a^2 \varepsilon_1 - b^2 \varepsilon_2}{a^2 - b^2} \right) (1 + \varepsilon_3) \\ &= (a^2 - b^2)(1 + \delta) \end{aligned} \quad (1.9)$$

gdzie $\delta = \frac{a^2 \varepsilon_1 - b^2 \varepsilon_2}{a^2 - b^2} (1 + \varepsilon_3) + \varepsilon_3$, a $\varepsilon_1, \varepsilon_2, \varepsilon_3$ są błędami wytworzonymi w poszczególnych działaniach arytmetycznych i zgodnie z zależnością (1.8) $|\varepsilon_i| \leq 2^{-t}$ ($i = 1, 2, 3$). Jeśli a^2 jest odpowiednio bliskie b^2 , a ε_1 i ε_2 mają przeciwne znaki, to błąd względny δ wyniku otrzymanego algorytmem $A1$ może być dowolnie duży. Nie jest tak w przypadku drugiego algorytmu, gdyż

$$\begin{aligned} fl((a - b)(a + b)) &= ((a - b)(1 + \varepsilon_1)(a + b)(1 + \varepsilon_2))(1 + \varepsilon_3) \\ &= (a^2 - b^2)(1 + \delta) \end{aligned} \quad (1.10)$$

gdzie $|\delta| \leq |\varepsilon_1| + |\varepsilon_2| + |\varepsilon_3|$ i błąd względny δ jest zawsze nie większy od $3 \cdot 2^{-t}$ (tam gdzie nie będzie to istotne, nie będziemy podkreślali, że jest to oszacowanie w sensie $\leq$).

UWAGA: Przy analizie obu algorytmów używaliśmy tych samych oznaczeń ε_i dla różnych błędów, np. ε_1 we wzorze (1.9) oznaczał błąd wytworzony przy mnożeniu $a \times a$, a w (1.10) przy odejmowaniu $a - b$. Mamy nadzieję, że nie będzie to dalej prowadziło do nieporozumień. $\square$

Przytoczony wyżej przykład pokazuje, jak istotnie różne własności numeryczne mogą mieć algorytmy równoważne w sensie klasycznej arytmetyki. Pokazuje również, że utrata dokładności wyniku wcale nie musi wiązać się z wielką liczbą zaokrągleń w trakcie obliczeń.

Ze wzoru (1.7) i przedstawionej dalej ilustracji dodawania liczb w t -cyfrowej arytmetyce zmiennopozycyjnej wynika, że dla argumentów a i b spełniających warunek $|b| \leq \frac{1}{2} \cdot 2^{-t} |a|$ zachodzi zależność

$$fl(a+b) \equiv a$$

Konsekwencje tego faktu często są dosyć zaskakujące.

PRZYKŁAD 1.7

Niech f będzie funkcją różniczkowalną w punkcie $x = 1$. Aproksymując pochodną ilorazem różnicowym

$$\frac{f(1+h)-f(1)}{h} \quad (1.11)$$

w arytmetyce klasycznej (dokładnej) dostajemy dla malejących h coraz lepsze przybliżenia $f'(1)$. Obliczając wyrażenie (1.11) w t -cyfrowej arytmetyce zmiennopozycyjnej mamy

$$fl\left(\frac{f(1+h)-f(1)}{h}\right) = fl\left(\frac{f(fl(1+h))-f(1)}{h}\right)$$

skąd wynika, że dla $0 < |h| < \frac{1}{2} \cdot 2^{-t}$ dostajemy dokładnie zero, gdyż dla tych wartości h jest $fl(1+h) \equiv 1$. A więc zamiast coraz lepszych przybliżeń pochodnej dla malejących h , obliczając wielkość (1.11) numerycznie, otrzymujemy od pewnej chwili te same wartości, nie mające żadnego związku z aproksymowaną wielkością, ani nawet z funkcją f . W przeciwieństwie do arytmetyki klasycznej nie możemy zatem dowolnie blisko aproksymować pochodnej ilorazem różnicowym. $\square$

Algorytmy numerycznie poprawne

1.4

Zastanówmy się teraz, jak dokładny może być dla danego zadania wynik obliczony w arytmetyce zmiennopozycyjnej. W ogólnym przypadku zamiast zadania o dokładnych danych numerycznie będziemy rozwiązywali zadanie o danych zaburzonych przez błędy reprezentacji. Podobnie, gdybyśmy nawet znali dokładne rozwiązanie dla tych danych, to w arytmetyce fl byłoby ono reprezentowane na ogół tylko w sposób przybliżony. Z tych względów *za numerycznie najwyższej jakości uznamy takie algorytmy, dla których obliczone rozwiązanie jest nieco zaburzonym rozwiązaniem (dokładnym) zadania o nieco zaburzonych danych*. Algorytmy spełniające powyższy postulat* nazywamy *numerycznie poprawnymi*. Przez „nieco zabu-

* Formułę tę przypisuje się W. Kahanowi, wydaje się jednak, że wcześniej sformułował ją J. Ortega.

rzony” rozumiemy tu zaburzenie na poziomie błędu reprezentacji. Chcemy także, ze względów praktycznych, aby omawiana własność algorytmu dotyczyła nie jednego ustalonego zadania, lecz całej możliwie szerokiej klasy zadań.

Sformalizowanie definicji algorytmu numerycznie poprawnego stwarza pewne trudności. Wątpliwości mogą powstać już przy ustaleniu, czym są dane zadania i to nawet zadania bardzo prostego.

PRZYKŁAD 1.8

Obliczając w arytmetyce fl wartość wielomianu dwu zmiennych $w(x, y; 1, 4) = x^2 + 4y^2$ o współczynnikach 1 i 4, otrzymujemy

$$\begin{aligned} fl(x^2 + 4y^2) &= (x \times x (1 + \varepsilon_1) + 4y \times y (1 + \varepsilon_2)) (1 + \varepsilon_3) \\ &= 1 (1 + \eta_1) x^2 + 4 (1 + \eta_2) y^2 \\ &= (x (1 + \delta_1))^2 + 4 (y (1 + \delta_2))^2 \end{aligned}$$

gdzie

$$\begin{aligned} |\varepsilon_i| &\leq 2^{-t} \\ 1 + \eta_1 &\stackrel{df}{=} (1 + \varepsilon_1) (1 + \varepsilon_3), \quad |\eta_1| \leq 2 \cdot 2^{-t} \\ 1 + \eta_2 &\stackrel{df}{=} (1 + \varepsilon_2) (1 + \varepsilon_3), \quad |\eta_2| \leq 2 \cdot 2^{-t} \\ 1 + \delta_1 &\stackrel{df}{=} \sqrt{1 + \eta_1}, \quad |\delta_1| \leq 2^{-t}, \quad 1 + \delta_2 \stackrel{df}{=} \sqrt{1 + \eta_2}, \quad |\delta_2| \leq 2^{-t} \end{aligned}$$

Otrzymany wynik możemy różnie interpretować, na przykład jako dokładną wartość wielomianu o nieco zaburzonych współczynnikach, bo

$$fl(w(x, y; 1, 4)) = w(x, y; 1(1 + \eta_1), 4(1 + \eta_2))$$

bądź jako dokładną wartość wielomianu dla nieco zaburzonych argumentów, gdyż

$$fl(w(x, y; 1, 4)) = w(x(1 + \delta_1), y(1 + \delta_2); 1, 4)$$

Jeśli za dane zadania przyjmiemy tylko współczynniki bądź tylko wartości argumentów, to w obu przypadkach algorytm jest ze względu na te dane numerycznie poprawny. Zauważmy przy okazji, że analizując algorytm nie uwzględniliśmy błędu wytworzonego przy mnożeniu przez 4. Ogólnie, mnożenie (dzielenie) przez całkowitą potęgę 2 w arytmetyce dwójkowej jest wykonywane dokładnie, gdyż jest równoważne tylko zmianie cechy drugiego argumentu. $\square$

Własność numerycznej poprawności algorytmów będziemy wykazywali na ogół przez konstrukcję pozornych zaburzeń danych równoważnych wytworzonym błędom zaokrągleń. Sprowadza się to, jak w poprzednich przykładach, do wykazania, że błędy wytworzone w poszczególnych działaniach algorytmu są równoważne skonstruowanym zaburzeniom danych początkowych. Z tego względu pewna dowolność w określeniu danych zadania jest czynnikiem pozytywnym, pozwalając często na wygodniejsze „przyczepianie” błędów do odpowiednio wybranych danych.

Jak poprzednio, ograniczamy się do zadań, polegających na obliczaniu dla danych $d = (d_1, d_2, \dots, d_n)$ wyniku $w = (w_1, w_2, \dots, w_m)$

$$w = \varphi(d)$$

gdzie φ jest ciągłym odwzorowaniem, $\varphi: R_d \rightarrow R_w$, określonym na podzbiorze $D_0 \subset R_d$ o wartościach w R_w , a R_d i R_w są skończone wymiarowymi unormowanymi przestrzeniami kartezjańskimi.

Będziemy mówili (por. praca [41]) o algorytmie A w klasie zadań $\{\varphi; D\}$, jeśli A oznacza algorytm obliczania wyniku $w = \varphi(d)$ dla dowolnych danych $d \in D \subset D_0$. Mamy tu na myśli dokładną realizację działań, tzn. arytmetykę klasyczną. Natomiast przez $fl(A(d))$ rozumiemy realizację algorytmu A w arytmetyce fl polegającą na

(a) zastąpieniu wielkości liczbowych występujących w A ich reprezentacjami zmiennopozycyjnymi;

(b) wykonywaniu operacji arytmetycznych (działań, obliczania wartości funkcji standardowych) w arytmetyce fl , a więc na ogół w sposób przybliżony.

Może się zdarzyć, że realizacja algorytmu A dla konkretnych danych $d \in D$ możliwa jest tylko przy założeniu dostatecznie silnej arytmetyki, tzn. dla t nie mniejszych od pewnego t_0 .

Zakładamy, że dane $d \in D$ i wyniki $w = \varphi(d)$ są reprezentowane w przestrzeniach R_d i R_w z błędami nie większymi odpowiednio od ϱ_d i ϱ_w , tzn.

$$\|d - rd(d)\| \leq \varrho_d \|d\| \quad \text{ i } \quad \|w - rd(w)\| \leq \varrho_w \|w\|$$

Stałe ϱ_d i ϱ_w są na ogół niewielkimi krotnościami 2^{-t} (zależnie od normy przestrzeni).

DEFINICJA 1.2

Algorytm A nazywamy *numerycznie poprawnym* w klasie zadań $\{\varphi; D\}$, jeśli istnieją stałe K_d, K_w takie, że dla każdego $d \in D$, dla dostatecznie silnej arytmetyki istnieje $\tilde{d} \in D_0$ takie, że

$$\|d - \tilde{d}\| \leq \varrho_d K_d \|d\| \tag{1.12}$$

$$\|\varphi(\tilde{d}) - fl(A(d))\| \leq \varrho_w K_w \|\varphi(\tilde{d})\| \tag{1.13} \quad \blacksquare$$

Dla dowolnego zadania rozważanej klasy, tj. zadania wyznaczenia elementu $\varphi(d)$ rozwiązanie $fl(A(d))$, obliczone algorytmem numerycznie poprawnym, różni się od rozwiązania dokładnego $\varphi(\tilde{d})$ co najwyżej K_w razy więcej niż poziom błędu jego reprezentacji. Jednocześnie $\varphi(\tilde{d})$ jest dokładnym rozwiązaniem zadania o danych $\tilde{d}$ różniących się od danych początkowych d co najwyżej K_d razy więcej niż poziom błędu ich reprezentacji.

Stałe K_d i K_w nazywamy *wskaźnikami kumulacji* algorytmu A w klasie zadań $\{\varphi; D\}$. Muszą być one dobrane tak, aby nierówności (1.12) i (1.13) zachodziły dla dowolnych danych rozważanej klasy. Chcielibyśmy jednocześnie, aby wartości

K_d i K_w były możliwie realistyczne, tzn. bliskie minimalnych. Stanowią one wtedy dobrą charakterystykę jakości algorytmu numerycznie poprawnego. Najlepsze ze znanych algorytmów mają wskaźniki kumulacji rzędu jedności. Istnieją jednak algorytmy, które spełniają definicję 1.2 z tak wielkimi stałymi K_d i K_w , że dla uzyskania nierówności $\varrho_d K_d < 1$ i $\varrho_w K_w < 1$ należałoby stosować szalenie silną, nieosiągalną w praktyce arytmetykę. Wydaje nam się więc, że w praktyce obliczeniowej są użyteczne raczej algorytmy poprawne ze wskaźnikami kumulacji tego rzędu co np. liczba działań arytmetycznych potrzebnych do rozwiązania danego zadania, czy jej niewielka potęga.

W wielu zadaniach nie ma sensu traktowanie wszystkich danych czy wyników jako elementu jednej przestrzeni unormowanej. Często znacznie bardziej naturalne jest rozważanie przestrzeni danych i wyników jako iloczynów kartezjańskich odpowiednich przestrzeni unormowanych. Na przykład dla układu liniowego $Ax = b$ byłoby rzeczą sztuczną traktować $n^2 + n$ liczb, elementów macierzy A i wektora b , jako elementu $(n^2 + n)$ -wymiarowej przestrzeni. Definicję algorytmu numerycznie poprawnego należy wtedy zmodyfikować zastępując (1.12) i (1.13) układem analogicznych nierówności odpowiadających składowym danych i wyników w odpowiednich produktach kartezjańskich (patrz praca [41]). Chcemy tu podkreślić, że określenie struktury danych i wyników jest bardzo istotne dla badania numerycznej poprawności algorytmu. Dokładniej wyjaśnimy to na jednym z dalszych przykładów.

Przedstawiony przez nas model jest pewnym uproszczeniem modelu zaproponowanego przez Kiełbasińskiego w pracy [41]. Nie obejmuje on dowolnych zadań pojawiających się w praktyce obliczeniowej, mimo to wydaje nam się wystarczająco ogólny. Inne próby sformalizowania pojęcia numerycznej poprawności algorytmu można znaleźć w pracach Babuśki [4], Wojewodina [78] czy Woźniakowskiego [82]. Warto też wspomnieć, że J. H. Wilkinson, któremu głównie zawdzięczamy rozwój analizy błędów w metodach numerycznych, wyjaśnia wprowadzane pojęcia [80], [81] jedynie na przykładach unikając ścisłych definicji.

PRZYKŁAD 1.9

Zbadajmy numeryczną poprawność algorytmu obliczania iloczynu skalarnego $S(a, b) = \sum_{i=1}^n a_i b_i$. Dane $a = (a_1, a_2, \dots, a_n)$ i $b = (b_1, b_2, \dots, b_n)$ traktujemy jako element iloczynu kartezjańskiego $R^n \times R^n$, gdzie R^n jest n -wymiarową przestrzenią rzeczywistą z normą euklidesową $\|a\| \stackrel{\text{def}}{=} \sqrt{\sum_{i=1}^n a_i^2}$. A zatem $S: R^n \times R^n \rightarrow R$.

Rozpatrywany algorytm ma w Algolu następującą postać

$S := 0;$

for $i := 1$ **step** 1 **until** n **do**

$S := S + a_i \times b_i;$

Jego zmiennopozycyjna realizacja $fl(A(a, b))$ będzie przebiegała następująco:

(a) dane a_i, b_i zostaną zastąpione ich reprezentacjami zmiennopozycyjnymi

$$\begin{aligned}\hat{a}_i &= rd(a_i) = a_i(1 + \alpha_i) \\ \hat{b}_i &= rd(b_i) = b_i(1 + \beta_i)\end{aligned}\quad (1.14)$$

gdzie $|\alpha_i|, |\beta_i| \leq 2^{-t}$ ($i = 1, 2, \dots, n$);

(b) działania arytmetyczne będą wykonywane w arytmetyce fl , a więc w sposób przybliżony. Dokładniej

$$\begin{aligned}fl(A(a, b)) &= (\dots (\hat{a}_1 \hat{b}_1(1 + \varepsilon_1) + \hat{a}_2 \hat{b}_2(1 + \varepsilon_2))(1 + \delta_2) + \\ &+ \dots + \hat{a}_n \hat{b}_n(1 + \varepsilon_n))(1 + \delta_n)\end{aligned}$$

gdzie ε_i są błędami względnymi wytworzonymi przy obliczaniu kolejnych składników $\hat{a}_i \hat{b}_i$, zaś δ_i — przy ich sumowaniu, $|\varepsilon_i|, |\delta_i| \leq 2^{-t}$. Uwzględniając zależności (1.14) dostajemy po przekształceniach

$$fl(A(a, b)) = \sum_{i=1}^n a_i(1 + \alpha_i) b_i(1 + E_i) \quad (1.15)$$

gdzie

$$1 + E_i \stackrel{\text{def}}{=} (1 + \beta_i)(1 + \varepsilon_i) \prod_{j=i}^n (1 + \delta_j), \quad \delta_1 = 0, \quad |E_i| \leq (n - i + 3) \cdot 2^{-t}$$

Obliczony wynik (1.15) możemy interpretować np. jako dokładny wynik (wskaźnik kumulacji $K_w = 0$) dla trochę zaburzonych danych

$$\begin{aligned}\tilde{a} &= (a_1(1 + \alpha_1), a_2(1 + \alpha_2), \dots, a_n(1 + \alpha_n)) \\ \tilde{b} &= (b_1(1 + E_1), b_2(1 + E_2), \dots, b_n(1 + E_n))\end{aligned}$$

takich, że

$$\begin{aligned}\|a - \tilde{a}\| &\leq 2^{-t} \|a\|, \quad (K_{a_i} = 1) \\ \|b - \tilde{b}\| &\leq (n + 1) 2^{-t} \|b\|, \quad (K_{a_2} = n + 1)\end{aligned}$$

Oczywiście błędy wytworzone w algorytmie można interpretować jako pozorne zaburzenia tylko składowych wektora a , czy też „podzielić” je dowolnie inaczej między a i b .

Zauważmy, że pominięcie błędów reprezentacji danych (punkt (a)) spowodowałoby po prostu zmniejszenie wskaźników kumulacji K_{a_i}, K_{a_2} o jeden. Często mamy do czynienia właśnie z taką sytuacją, gdy dane początkowe rozważanego algorytmu są swoimi własnymi reprezentacjami. Jest tak na przykład zawsze wtedy, gdy dane te są wynikami wcześniejszych obliczeń. Z tego względu w *analizie numerycznej poprawności* będziemy dalej zakładali, że dane są reprezentowane dokładnie. Wystarczy pamiętać, że dopuszczenie danych dowolnych wiąże się z powiększeniem K_4 o 1. $\square$

PRZYKŁAD 1.10

Dla $a > 0$ obliczamy przybliżenie $\sqrt{a}$ algorytmem Herona*, tzn. konstruujemy ciąg $\{x_i\}$ zgodnie ze wzorami

$$x_1 = \begin{cases} a, & \text{jeśli } a \geq 1 \\ 1, & \text{jeśli } a < 1 \end{cases}$$

$$x_{i+1} = \frac{1}{2} \left(x_i + \frac{a}{x_i} \right), \quad i = 1, 2, \dots$$

Ciąg ten jest zbieżny do $\sqrt{a}$, gdyż

$$\begin{aligned} 0 \leq x_{i+1} - \sqrt{a} &= \frac{1}{2} \left(x_i + \frac{a}{x_i} - 2\sqrt{a} \right) = \\ &= \frac{(x_i - \sqrt{a})^2}{2x_i} \leq \frac{1}{2} (x_i - \sqrt{a}) \leq \dots \leq \frac{1}{2^i} (x_1 - \sqrt{a}) \end{aligned} \quad (1.16)$$

Oznaczmy przez $\tilde{x}_i$ wielkości otrzymywane przy zmiennopozycyjnej realizacji algorytmu dla $a = rd(a)$. Zachodzą dla nich zależności

$$\tilde{x}_1 = x_1$$

$$\tilde{x}_{i+1} = fl \left(\frac{1}{2} \left(\tilde{x}_i + \frac{a}{\tilde{x}_i} \right) \right) = \frac{1}{2} \left(\tilde{x}_i + \frac{a}{\tilde{x}_i} (1 + \varepsilon_i) \right) (1 + \delta_i),$$

gdzie $|\varepsilon_i|, |\delta_i| \leq 2^{-t}$

Podobnie jak we wzorze (1.16) zbadajmy zachowanie się ciągu błędów $\{\tilde{x}_i - \sqrt{a}\}$

$$\begin{aligned} \tilde{x}_{i+1} - \sqrt{a} &= \left(\frac{1}{2} \left(\tilde{x}_i + \frac{a}{\tilde{x}_i} \right) - \sqrt{a} \right) (1 + \delta_i) + \left(\delta_i \sqrt{a} + \frac{a\varepsilon_i}{2\tilde{x}_i} \right) = \\ &= \frac{(\tilde{x}_i - \sqrt{a})^2}{2\tilde{x}_i} (1 + \delta_i) + \xi_i \end{aligned} \quad (1.17)$$

gdzie

$$\xi_i = \delta_i \sqrt{a} + \frac{a\varepsilon_i}{2\tilde{x}_i}, \quad |\xi_i| \leq 2^{-t} \left(\sqrt{a} + \frac{a}{2|\tilde{x}_i|} \right)$$

Zauważmy, że $\tilde{x}_i \geq \sqrt{a}$ (sprawdzenie tego faktu pozostawiamy jako ćwiczenie).

Stąd

$$|\xi_i| \leq \frac{3}{2} 2^{-t} \sqrt{a}$$

* Algorytm Herona jest to metoda Newtona (omawiana w rozdz. 7 części drugiej) dla funkcji $f(x) = x^2 - a$.

i uwzględniając to oszacowanie w równości (1.17) otrzymujemy

$$|\tilde{x}_{i+1} - \sqrt{a}| \leq \frac{(\tilde{x}_i - \sqrt{a})^2}{2\sqrt{a}} + \frac{3}{2} \cdot 2^{-i} \sqrt{a} \quad (1.18)$$

Z zależności (1.17) widać, że ciąg $\{\tilde{x}_i - \sqrt{a}\}$, w odróżnieniu od $\{x_i - \sqrt{a}\}$ nie musi być zbieżny do zera. Co więcej nie musi być w ogóle zbieżny, bo zaburzenia δ_i i ε_i są jak gdyby wielkościami losowymi. Oszacujmy zatem granicę górną $e = \limsup_{i \rightarrow \infty} |\tilde{x}_i - \sqrt{a}|$. Korzystając z zależności (1.18) otrzymujemy dla e nierówność kwadratową

$$e^2 - 2\sqrt{a}e + 3a \cdot 2^{-i} \geq 0$$

a stąd

$$e \leq \frac{3}{2} 2^{-i} \sqrt{a}$$

Zatem dla dowolnie małego $\varepsilon > 0$ istnieje taki indeks k_0 , że dla $k > k_0$

$$|\tilde{x}_k - \sqrt{a}| \leq \varepsilon + \frac{3}{2} \cdot 2^{-i} \sqrt{a}$$

Biorąc np. $\varepsilon = 2^{-2i} \sqrt{a}$ mamy

$$|\tilde{x}_k - \sqrt{a}| \leq \frac{3}{2} \cdot 2^{-i} \sqrt{a}$$

co oznacza, że obliczone $\tilde{x}_k$ jest trochę (półtora raza więcej niż błąd reprezentacji) zaburzonym dokładnym rozwiązaniem zadania o dokładnych danych. Pokazaliśmy zatem nie tylko, że algorytm jest numerycznie poprawny, ale, co więcej, że obliczony wynik ma zawsze bardzo dużą dokładność względną. $\square$

Algorytmy numerycznie stabilne

1.5

Jak pokazaliśmy poprzednio, przybliżona reprezentacja numeryczna danych i wyników powoduje, że rozwiązanie zadania na ogół obciążone jest pewnym błędem, niezależnym od metody jego rozwiązywania. Chcemy zbadać teraz, jaki jest poziom błędu rozwiązania spowodowanego zaburzeniem danych i wyniku na poziomie błędu ich reprezentacji. Wydaje się to celowe także z tego powodu, że nawet jeśli dane i wyniki są reprezentowane dokładnie, to w algorytmach numerycznie najwyższej jakości (tzn. numerycznie poprawnych) błędy wytworzone w poszczególnych działaniach możemy interpretować jako zaburzenia danych i wyniku właśnie na poziomie błędu ich reprezentacji.

Rozpatrujemy zadanie obliczenia dla danych d wyniku $w = \varphi(d)$. Niech $\hat{d}$ będą danymi zaburzonymi na poziomie błędu reprezentacji, a $\tilde{w}$ zaburzonym na poziomie błędu reprezentacji dokładnym rozwiązaniem zadania dla tych danych.

Dokładniej

$$\begin{aligned} w &= \varphi(d), & \|d - \hat{d}\| &\leq \varrho_d \|d\| \\ \hat{w} &= \varphi(\hat{d}), & \|\hat{w} - \tilde{w}\| &\leq \varrho_w \|\hat{w}\| \end{aligned} \quad (1.19)$$

Otrzymujemy stąd oszacowanie

$$\|w - \tilde{w}\| \leq \|w - \hat{w}\| + \|\hat{w} - \tilde{w}\| \leq (1 + \varrho_w) \max_{\hat{d}: \|d - \hat{d}\| \leq \varrho_d \cdot \|d\|} \|\varphi(d) - \varphi(\hat{d})\| + \varrho_w \|w\|$$

Podobnie jak w pracy [41] wielkość

$$P(d, \varphi) \stackrel{\text{def}}{=} \varrho_w \|w\| + \max_{\hat{d}: \|d - \hat{d}\| \leq \varrho_d \cdot \|d\|} \|\varphi(d) - \varphi(\hat{d})\|$$

będziemy nazywali *optymalnym poziomem błędu* rozwiązania $\varphi(d)$ w arytmetyce *fl*.

Wprowadzamy także za Kiełbasińskim [41] następującą definicję.

DEFINICJA 1.3

Algorytm A nazywamy *numerycznie stabilnym* w klasie zadań $\{\varphi; D\}$, jeśli istnieje stała K taka, że dla każdego $d \in D$, dla dostatecznie silnej arytmetyki zachodzi nierówność

$$\|\varphi(d) - fl(A(d))\| \leq K \cdot P(d, \varphi) \quad (1.20) \blacksquare$$

Możemy więc powiedzieć, że algorytm numerycznie stabilny to taki, który gwarantuje uzyskanie rozwiązania (dla dowolnego zadania rozważanej klasy) z błędem tego samego rzędu, co optymalny poziom błędu rozwiązania danego zadania. Możemy powiedzieć jeszcze prościej, że błąd w algorytmie numerycznie stabilnym jest na poziomie „nieuniknionego” błędu rozwiązania wynikającego z przybliżonej reprezentacji danych i wyniku.

Podobnie jak w definicji algorytmu numerycznie poprawnego zakładamy milcząco, że wskaźnik stabilności K nie jest zbyt duży. Jego wielkość może służyć do oceny jakości algorytmu. Im mniejsze jest K , tym lepszy algorytm.

Z porównania zależności (1.12), (1.13) z (1.19) wynika, że każdy algorytm numerycznie poprawny jest algorytmem numerycznie stabilnym. Własności te nie są jednak równoważne. Przykłady algorytmów numerycznie stabilnych nie będących numerycznie poprawnymi podajemy w dalszych rozdziałach.

Podkreślimy jeszcze raz, że numeryczna poprawność (z niewielkimi stałymi K_d i K_w) jest maksymalną własnością, jakiej możemy oczekiwać od algorytmu. Ze względu na błąd reprezentacji danych i wyniku jedynie przez przypadek moglibyśmy uzyskać istotnie lepsze rozwiązanie. Z kolei numeryczna stabilność (z niezbyt dużym wskaźnikiem K) jest minimalną własnością, jakiej powinniśmy domagać się od algorytmu. Rezygnacja z własności numerycznej stabilności oznaczałaby rezygnację z ograniczonej strat dokładności w stosunku do tego, co możemy uzyskać w stosowanej arytmetyce.

Jeśli $w = \varphi(d) \neq 0$, to z nierówności (1.20) możemy wyprowadzić oszacowanie błędu względnego rozwiązania $fl(A(d))$ obliczonego algorytmem numerycznie stabilnym

$$\frac{\|\varphi(d) - fl(A(d))\|}{\|\varphi(d)\|} \leq K(\text{cond}(d) \varrho_d + \varrho_w) \quad (1.21)$$

Wielkość

$$\text{cond}(d) \stackrel{\text{def}}{=} \frac{\max_{\hat{d}: \|\hat{d} - d\| \leq \varrho_d \cdot \|d\|} \|\varphi(d) - \varphi(\hat{d})\|}{\varrho_d \|\varphi(d)\|}$$

przemnożona przez ϱ_d określa maksymalną względną zmianę rozwiązania $\varphi(d)$ przy zaburzeniu danych na poziomie błędu reprezentacji. Jest to zatem miara wrażliwości zadania na niewielkie względne zmiany danych i zgodnie z poprzednio wprowadzoną definicją możemy $\text{cond}(d)$ uważać za wskaźnik uwarunkowania zadania.

Wyjaśnialiśmy już przy omawianiu numerycznej poprawności algorytmów, że rozważany przez nas model obejmuje tylko najprostszy przypadek, w którym wynik traktujemy jako element jednej przestrzeni unormowanej, a nie rozbijamy go na składowe należące do różnych przestrzeni, będących czynnikami odpowiedniego iloczynu kartezjańskiego. W praktyce często bardziej użyteczny jest ten ogólniejszy model. Można wtedy określić optymalny poziom błędu dla każdej składowej wyniku. W przypadku numerycznej stabilności wszystkich odpowiednich składowych algorytmu uznajemy go za numerycznie stabilny. Analogicznie możemy definiować wskaźniki uwarunkowania dla składowych rozwiązań względem składowych danych zamiast globalnego wskaźnika $\text{cond}(d)$.

W każdym przypadku widać (por. nierówność (1.20)), że jeśli zadanie jest dobrze uwarunkowane (tzn. $\text{cond}(d)$ jest niewielkie), to rozwiązując je algorytmem numerycznie stabilnym (tym bardziej numerycznie poprawnym) otrzymujemy rozwiązanie z niewielkim błędem względnym.

Możemy przejść teraz do omówienia następującego przykładu.

PRZYKŁAD 1.11

W arytmetyce $fl(t = 24)$ wyznaczano pierwiastki x_1, x_2 równania kwadratowego $x^2 - 2px + q = 0$ o współczynnikach p, q ze zbioru $D = \{(p, q): p \neq 0, q \neq 0, p^2 - q > 0\}$ dwoma algorytmami

– „szkolnym”

$$x_1 := p + \text{sqrt}(p \times p - q);$$

$$x_2 := p - \text{sqrt}(p \times p - q);$$

– zmodyfikowanym

```

if  $p \geq 0$  then
begin
 $x_1 := p + \text{sqrt}(p \times p - q)$ ;
 $x_2 := q/x_1$ 
end else
begin
 $x_2 := p - \text{sqrt}(p \times p - q)$ ;
 $x_1 := q/x_2$ 
end
    
```

Dla różnych wartości p i q otrzymano następujące przybliżenia pierwiastków (tabl. 1.1).

TABLICA 1.1

p	q	algorytm „szkolny”	algorytm zmodyfikowany	dokładne pierwiastki
-0.435001	$0.174_{10}-5$	$x_1 = -0.208616_{10}-5$ $x_2 = -0.870000$	$x_1 = -0.200000_{10}-5$ $x_2 = -0.870000$	$x_1 = -0.2_{10}-5$ $x_2 = -0.87$
$1_{10}5$	2	$x_1 = 0.200000_{10}6$ $x_2 = 0.000000$	$x_1 = 0.200000_{10}6$ $x_2 = 0.100000_{10}-4$	$x_1 = 0.200000_{10}6^*$ $x_2 = 0.100000_{10}-4^*$
$0.499981_{10}3$	$0.199992_{10}1$	$x_1 = 0.999960_{10}3$ $x_2 = 0.195313_{10}-2$	$x_1 = 0.999960_{10}3$ $x_2 = 0.200000_{10}-2$	$x_1 = 0.99996_{10}3$ $x_2 = 0.2_{10}-2$

* Z dokładnością do sześciu cyfr.

Dlaczego we wskazanych przykładach jeden z pierwiastków obliczanych algorytmem „szkolnym” jest obarczony dużym błędem względnym?

Obliczając wartości wskaźników uwarunkowania cond_p^k i cond_q^k ($k = 1, 2$), wyprowadzonych w przykładzie 1.4, widzimy, że dla podanych w tablicy wartości p i q zadanie wyznaczania każdego z zer rozważanego trójmianu jest bardzo dobrze uwarunkowane. Duże względne błędy wyników obliczonych algorytmem „szkolnym” muszą być zatem spowodowane jego złą jakością numeryczną.

Przeanalizujmy ten algorytm wspólnie dla obu pierwiastków. W dalszych wzorach używamy znaku „ $\pm$ ” pamiętając, że „ $+$ ” odpowiada wyznaczeniu x_1 , a „ $-$ ” x_2

$$fl(p \pm \sqrt{p^2 - q}) = (p \pm \sqrt{(p^2(1 + \varepsilon_1) - q)(1 + \varepsilon_2)(1 + \varepsilon_3)})(1 + \varepsilon_4) \quad (1.22)$$

gdzie $|\varepsilon_i| \leq 2^{-t}$, $i = 1, 2, 3, 4$ (zakładamy tutaj dla uproszczenia, że dla $x = rd(x) \geq 0$ jest spełniona zależność $fl(\sqrt{x}) = \sqrt{x}(1 + \varepsilon)$, $|\varepsilon| \leq 2^{-t}$; por. p. 1.7).

Przekształcenia, które teraz wykonamy, mają dwa cele. Chcemy pokazać, że dla pewnych danych wynik (dokładniej, jeden z pierwiastków) obliczony algorytmem „szkolnym” może być obarczony dowolnie dużym błędem względnym. Jednocześnie chcemy zbadać, jak można zmodyfikować ten algorytm, aby był numerycznie poprawny dla dowolnych danych ze zbioru D .

Ze wzoru (1.22) otrzymujemy

$$\begin{aligned} fl(p \pm \sqrt{p^2 - q}) &= \left(p \sqrt{1 + \varepsilon_1} \frac{1}{\sqrt{1 + \varepsilon_1} \sqrt{1 + \varepsilon_2} (1 + \varepsilon_3)} \pm \right. \\ &\quad \left. \pm \sqrt{(p \sqrt{1 + \varepsilon_1})^2 - q} \right) \sqrt{1 + \varepsilon_2} (1 + \varepsilon_3) (1 + \varepsilon_4) = \\ &= (p (1 + \eta_1) (1 + \eta_2) \pm \sqrt{(p (1 + \eta_1))^2 - q}) (1 + \eta_3) \end{aligned}$$

gdzie

$$\begin{aligned} 1 + \eta_1 &= \sqrt{1 + \varepsilon_1} \quad \text{i} \quad |\eta_1| \leq \frac{1}{2} \cdot 2^{-t} \\ 1 + \eta_2 &= \frac{1}{\sqrt{1 + \varepsilon_1} \sqrt{1 + \varepsilon_2} (1 + \varepsilon_3)} \quad \text{i} \quad |\eta_2| \leq 2 \cdot 2^{-t} \\ 1 + \eta_3 &= \sqrt{1 + \varepsilon_2} (1 + \varepsilon_3) (1 + \varepsilon_4) \quad \text{i} \quad |\eta_3| \leq \frac{5}{2} \cdot 2^{-t} \end{aligned}$$

A stąd

$$\begin{aligned} fl(p \pm \sqrt{p^2 - q}) &= (\tilde{p} \pm \sqrt{\tilde{p}^2 - q}) \left(1 + \frac{\tilde{p} \eta_2}{\tilde{p} \pm \sqrt{\tilde{p}^2 - q}} \right) (1 + \eta_3) = \\ &= (\tilde{p} \pm \sqrt{\tilde{p}^2 - q}) (1 + \eta_4) \end{aligned} \quad (1.23)$$

gdzie $\tilde{p} = p (1 + \eta_1)$, a $1 + \eta_4$ jest iloczynem dwóch ostatnich czynników po prawej stronie równości (1.23) i $|\eta_4|$ ma oszacowanie

$$|\eta_4| \leq \left(\frac{2 |\tilde{p}|}{|\tilde{p} \pm \sqrt{\tilde{p}^2 - q}|} + \frac{5}{2} \right) \cdot 2^{-t} \quad (1.24)$$

Jeśli $p > 0$, to dla pierwiastka x_1 mamy $|\eta_4| \leq \frac{9}{2} \cdot 2^{-t}$. Z zależności (1.23) wynika wtedy, że algorytm „szkolny” wyznaczania x_1 jest w tej klasie zadań ($p > 0$) numerycznie poprawny, gdyż nieco zaburzony (pomnożony przez $1 + \eta_4$) obliczony wynik jest dokładnym rozwiązaniem zadania o nieco zaburzonych danych (dokładniej, o zaburzonym współczynniku $\tilde{p} = p (1 + \eta_1)$ i dokładnym współczynniku q). Natomiast dla $p > 0$ oszacowanie błędu wytworzonego w arytmetyce fl przy wyznaczaniu algorytmem „szkolnym” drugiego pierwiastka x_2 może być dowolnie duże, gdyż dla $p > 0$ ułamek $\frac{|\tilde{p}|}{|\tilde{p} - \sqrt{\tilde{p}^2 - q}|} \rightarrow \infty$ przy $q/p^2 \rightarrow 0$.

Podobnie, dla $p < 0$ wielkość $|\eta_4|$ w przypadku wyznaczania x_1 może być dowolnie duża, natomiast dla x_2 jest nie większa niż $\frac{9}{2} \cdot 2^{-t}$.

Czytelnik zechce sprawdzić, że przytoczone wyniki testów potwierdzają przeprowadzoną tu analizę.

Widoczne jest teraz, dlaczego zmodyfikowany algorytm zapewnia numeryczną poprawność wyznaczania obu pierwiastków. Dla $p > 0$ ze wzoru (1.23) mamy

$$\tilde{x}_1 = fl(p + \sqrt{p^2 - q}) = (\tilde{p} + \sqrt{\tilde{p}^2 - q})(1 + \eta_4) \quad (1.25)$$

gdzie

$$\tilde{p} = p(1 + \eta_1), \quad |\eta_1| \leq \frac{1}{2} \cdot 2^{-t}, \quad \text{a} \quad |\eta_4| \leq \frac{9}{2} \cdot 2^{-t}$$

Obliczając x_2 dostajemy zatem

$$\tilde{x}_2 = fl(q/\tilde{x}_1) = \frac{q}{\tilde{x}_1} (1 + \varepsilon) = \frac{q}{\tilde{p} + \sqrt{\tilde{p}^2 - q}} \frac{1 + \varepsilon}{1 + \eta_4} = (\tilde{p} - \sqrt{\tilde{p}^2 - q})(1 + \eta_5) \quad (1.26)$$

gdzie

$$1 + \eta_5 = \frac{1 + \varepsilon}{1 + \eta_4} \quad \text{i} \quad |\varepsilon| \leq 2^{-t}$$

a stąd

$$|\eta_5| \leq \frac{11}{2} \cdot 2^{-t}$$

Dla $p < 0$ obliczone wyniki $\tilde{x}_1$ i $\tilde{x}_2$ będą znowu postaci (1.25) i (1.26) z innymi zaburzeniami η_4 i η_5 o oszacowaniach

$$|\eta_4| \leq \frac{11}{2} \cdot 2^{-t} \quad \text{i} \quad |\eta_5| \leq \frac{9}{2} \cdot 2^{-t}$$

Pokazaliśmy zatem, że zmodyfikowany algorytm rozwiązywania zadania

$$(x_1, x_2) = \varphi(p, q) = (p + \sqrt{p^2 - q}, p - \sqrt{p^2 - q})$$

jest numerycznie poprawny dla dowolnych danych $p \neq 0, q \neq 0, p^2 - q \geq 0$. Nieco zaburzony obliczony wynik jest bowiem dokładnym rozwiązaniem zadania o trochę zaburzonych danych. Ściślej

$$|p - \tilde{p}| \leq \frac{1}{2} \cdot 2^{-t} |p|$$

$$|\varphi(\tilde{p}, q) - \tilde{x}_i| \leq \frac{11}{2} \cdot 2^{-t} |\varphi_i(\tilde{p}, q)| \quad \text{dla} \quad i = 1, 2$$

gdzie $\varphi_i(p, q)$ są składowymi zadania $\varphi(p, q) = (\varphi_1(p, q), \varphi_2(p, q))$. Odpowiada to traktowaniu przestrzeni danych R_d i wyników R_w jako iloczynów kartezjańskich $R \times R$. Wspominaliśmy wcześniej, że numeryczna poprawność algorytmu może w istotny sposób zależeć od przyjętej struktury przestrzeni danych i wyników. Oczywiście przy tak określonej strukturze danych ($R_d = R_w = R \times R$) algorytm „szkolny” nie jest nawet stabilny, a tym bardziej numerycznie poprawny. Jeśli

jednak za przestrzeń wyników będziemy uważali $R_w = R^2$ z normą euklidesową, to algorytm „szkolny” jest numerycznie poprawny dla dowolnych danych ($p \neq 0$, $q \neq 0$, $p^2 - q \geq 0$). Z równości (1.23) mamy bowiem

$$\begin{aligned}\hat{x}_1 &= fl(p + \sqrt{p^2 - q}) = (\tilde{p} + \sqrt{\tilde{p}^2 - q}) \left(1 + \frac{\tilde{p}\eta_2}{\tilde{p} + \sqrt{\tilde{p}^2 - q}} + \eta_3\right) \\ \hat{x}_2 &= fl(p - \sqrt{p^2 - q}) = (\tilde{p} - \sqrt{\tilde{p}^2 - q}) \left(1 + \frac{\tilde{p}\eta'_2}{\tilde{p} - \sqrt{\tilde{p}^2 - q}} + \eta'_3\right)\end{aligned}$$

gdzie

$$\tilde{p} = p(1 + \eta_1), \quad |\eta_1| \leq \frac{1}{2} \cdot 2^{-t}$$

$$|\eta_2|, |\eta'_2| \leq 2 \cdot 2^{-t}, \quad |\eta_3|, |\eta'_3| \leq \frac{5}{2} \cdot 2^{-t}$$

A stąd

$$\begin{aligned}\frac{\|\varphi(\tilde{p}, q) - (\hat{x}_1, \hat{x}_2)\|}{\|\varphi(\tilde{p}, q)\|} &= \\ &= \left(\frac{\tilde{p}\eta_2 + (\tilde{p} + \sqrt{\tilde{p}^2 - q})\eta_3)^2 + (\tilde{p}\eta'_2 + (\tilde{p} - \sqrt{\tilde{p}^2 - q})\eta'_3)^2}{(\tilde{p} + \sqrt{\tilde{p}^2 - q})^2 + (\tilde{p} - \sqrt{\tilde{p}^2 - q})^2} \right)^{\frac{1}{2}} \leq \frac{11}{2} \cdot 2^{-t}\end{aligned}$$

Algorytm jest więc numerycznie poprawny i to z małą stałą kumulacyjną, $K_w = \frac{11}{2}$. Wynik jest dosyć zaskakujący. Zauważmy jednak, że w algorytmie „szkolnym” z dużym błędem względnym może być wyznaczony tylko mniejszy co do modułu pierwiastek. Ze wzoru (1.24) widzimy, że jest to możliwe jedynie wówczas, gdy $p^2 \gg q$, a wtedy wartość bezwzględna drugiego pierwiastka jest znacznie większa. Jeśli mierzymy błąd rozwiązania nie indywidualnie dla każdej składowej ($R_w = R \times R$), ale w normie euklidesowej dla całego wyniku ($R_w = R^2$), to nawet duże zaburzenie względne mniejszego pierwiastka jest nieistotne. $\square$

Mamy nadzieję, że powyższy przykład zwróci uwagę Czytelnika na potrzebę analizowania nawet prostych zadań. Przykład ten uczy, że często należy rozwiązywać zadania innymi algorytmami niż te, do których przywykliśmy w klasycznej arytmetyce. Ważne jest także, aby jakość numeryczna algorytmów, tzn. ich poprawność lub stabilność rozważać w normach odpowiadających sensowi fizycznemu czy technicznemu rozwiązywanego zadania.

Arytmetyki wyższej precyzji

1.6

W wielu maszynach cyfrowych istnieje możliwość korzystania z *arytmetyki podwójnej* (czy nawet wyższej) *precyzji*. Na reprezentację mantysy zarówno danych, jak i wyników działań arytmetycznych jest przeznaczonych wtedy nie t bitów, jak

w pojedynczej precyzji, ale $2t$ (czy nawet więcej) bitów. W arytmetyce podwójnej precyzji są spełnione zależności (1.5) i (1.8) na ogół z $|\varepsilon| \leq 2^{-2t}$. Analizy algorytmów w różnych arytmetykach różnią się najczęściej tylko oszacowaniami błędów reprezentacji liczb i błędów wytworzonych w działaniach.

Zależnie od konkretnej realizacji koszt działań arytmetycznych w podwójnej precyzji jest od kilku do kilkudziesięciu razy większy od kosztu wykonania tych działań w pojedynczej precyzji.

Dla niektórych zadań możliwe jest uzyskanie podwyższonej precyzji wyniku dzięki specjalnej (stosunkowo taniej) realizacji arytmetyki fl . Jedną z takich możliwości jest *algorytm Gilla-Möllera* [56] sumowania $\sum_{i=1}^n a_i$ z poprawkami. Zakładamy, że sumowanie to będzie się odbywało w arytmetyce fl z poprawnymi zaokrągleniami wyników działań. Algorytm Gilla-Möllera ma postać (por. praca [43])

```

u := p := 0;
for i := 1 step 1 until n do
begin
  v := <i-ty składnik sumy>;
  s := u + v;
  p := u - s + v + p;
  u := s
end;
z := s + p;

```

(u, p, v, s, z są zmiennymi rzeczywistymi a i, n całkowitymi). Zmiennej s są nadawane wartości kolejno obliczanych sum częściowych, a zmiennej p wartość poprawki sumowania. Jeśli $2^{-t}(n+4)^2 < 0.1$ (por. artykuł [43]), to obliczone z jest równe

$$z = \left(\sum_{i=1}^n a_i (1 + \varepsilon_i) \right) (1 + \varepsilon), \quad \text{gdzie} \quad |\varepsilon_i|, |\varepsilon| \leq 2^{-t} \quad (1.27)$$

Jeśli systematycznie stosujemy przedstawiony algorytm sumowania z poprawkami, to będziemy mówili (dla skrótu) o realizacji w arytmetyce $\tilde{fl}$.

Ze wzoru (1.27) wynika w szczególności, że

$$\tilde{fl} \left(\sum_{i=1}^n a_i b_i \right) = \sum_{i=1}^n a_i b_i (1 + \varepsilon_i)$$

gdzie $|\varepsilon_i| \leq 3 \cdot 2^{-t}$, a więc oszacowanie błędów jest niezależne od liczby sumowanych składników. W przeciwieństwie do realizacji arytmetyki fl (por. zależność

(1.15)) w arytmetyce $\tilde{fl}$ nie ma zatem w sumowaniu kumulacji błędów wytworzonych. Jeszcze dokładniejszy wynik można otrzymać w arytmetyce fl_2 . Rozumiemy przez nią obliczanie iloczynów $a_i \times b_i$ i ich sumowanie w podwójnej precyzji, a następnie reprezentowanie otrzymanego wyniku w pojedynczej precyzji. Można pokazać [80], że jeśli $\frac{3}{2}n \cdot 2^{-2t} < 0.1$, to

$$fl_2 \left(\sum_{i=1}^n a_i b_i \right) = \left(\sum_{i=1}^n a_i b_i (1 + \varepsilon_i) \right) (1 + \varepsilon) \quad (1.28)$$

gdzie

$$|\varepsilon_i| \leq \frac{3}{2} (n-i+2) \cdot 2^{-2t} \quad \text{dla } i = 1, 2, \dots, n, \quad \text{a } |\varepsilon| \leq 2^{-t}$$

Dla współczesnych maszyn cyfrowych ograniczenia wielkości n , dla których zachodzą równości (1.27) i (1.28), są nieistotne z praktycznego punktu widzenia. Na ogół znacznie silniejsze ograniczenia n wynikają z wielkości pamięci i czasu obliczeń.

Ilustrowanie przez nas większości wprowadzanych pojęć przykładami z iloczynem skalarnym i podawanie algorytmów jego dokładniejszego obliczania nie jest przypadkowe. Jest to bowiem jedno z najczęściej występujących zadań w praktyce obliczeniowej. Z tego też względu chcielibyśmy właśnie na przykładzie iloczynu skalarnego obliczania pokazać dwa różne sposoby szacowania błędu wytworzonego w algorytmie.

Z analizy wykonanej w przykładzie 1.9 oraz z równości (1.27) i (1.28) wynika, że obliczając numerycznie $S = \sum_{i=1}^n a_i b_i$ otrzymujemy wielkość $\tilde{S}$ równą

$$\tilde{S} = \left(\sum_{i=1}^n a_i b_i (1 + \varepsilon_i) \right) (1 + \varepsilon) \quad (1.29)$$

gdzie $|\varepsilon| \leq 2^{-t}$ oraz, zależnie od stosowanej arytmetyki,

$$|\varepsilon_i| \leq m_i \cdot 2^{-t}, \quad m_i = \begin{cases} n-i+1 & \text{dla } fl \\ 2 & \text{dla } \tilde{fl} \\ \frac{3}{2} (n-i+2) \cdot 2^{-t} & \text{dla } fl_2 \end{cases}$$

Obliczony iloczyn skalarny $\tilde{S}$ różni się więc od S o błąd e postaci

$$e = \sum_{i=1}^n a_i b_i \varepsilon_i + \tilde{S} \varepsilon / (1 + \varepsilon)$$

Otrzymujemy stąd tzw. oszacowanie „a priori” błędu wytworzonego

$$|e| \leq 2^{-t} \left(\sum_{i=1}^n |a_i b_i| m_i + \left| \sum_{i=1}^n a_i b_i \right| \right) \quad (1.30)$$

Oszacowanie to dla arytmetyki fl i $\tilde{fl}$ może być istotnie zawyżone ze względu na możliwość wystąpienia takich korelacji znaków składników sumy $\sum_{i=1}^n a_i b_i \varepsilon_i$, że $|\sum_{i=1}^n a_i b_i \varepsilon_i|$ będzie znacznie mniejsze od $\sum_{i=1}^n |a_i b_i \varepsilon_i|$.

Bardziej realistyczne oszacowania można uzyskać wykorzystując „na bieżąco” wielkości otrzymywane w trakcie realizacji obliczeń. Rozpatrzmy to dokładniej w przypadku arytmetyki fl .

Obliczając iloczyn skalarny algorytmem

```
S := 0;
for i := 1 step 1 until n do
begin
  ci := ai × bi;
  S := S + ci
end;
```

otrzymamy w fl wielkości $\tilde{c}_i$ i $\tilde{S}_i$ równe

$$\tilde{S}_0 = 0;$$

dla $i = 1, 2, \dots, n$

$$\tilde{c}_i = a_i b_i (1 + \varepsilon_i)$$

$$\tilde{S}_i = (\tilde{S}_{i-1} + \tilde{c}_i) (1 + \delta_i) = \tilde{S}_{i-1} + a_i b_i + \tilde{c}_i \frac{\varepsilon_i}{1 + \varepsilon_i} - \frac{\delta_i}{1 + \delta_i} \tilde{S}_i,$$

$$|c_i|, |\delta_i| \leq 2^{-t}$$

A więc

$$\tilde{S} = \tilde{S}_n = \sum_{i=1}^n a_i b_i + \sum_{i=1}^n \left(\tilde{c}_i \frac{\varepsilon_i}{1 + \varepsilon_i} + \frac{\delta_i}{1 + \delta_i} \tilde{S}_i \right)$$

i stąd możemy oszacować błąd wytworzony

$$\left| \tilde{S} - \sum_{i=1}^n a_i b_i \right| \leq 2^{-t} \sum_{i=1}^n (|\tilde{c}_i| + |\tilde{S}_i|)$$

Algorytm obliczania iloczynu skalarnego S (w arytmetyce fl) z jednoczesnym wyznaczaniem „na bieżąco” oszacowania e błędu wytworzonego można zapisać następująco

```
S := e := 0;
for i := 1 step 1 until n do
```

begin

$c := a_i \times b_i;$

$S := S + c;$

$e := e + |S| + |c|$

end;

$e := e \times 2 \uparrow (-t);$

Samo oszacowanie błędu e jest obliczane z dużą dokładnością względną jako suma dodatnich składników. Zauważmy bowiem, że

$$fl\left(\sum_{i=1}^n d_i\right) = \sum_{i=1}^n d_i(1+E_i) = \left(\sum_{i=1}^n d_i\right)(1+\delta)$$

gdzie

$$|E_i| \leq (n-i+1) \cdot 2^{-t} \quad \text{a} \quad \delta = \left(\sum_{i=1}^n d_i E_i\right) / \left(\sum_{i=1}^n d_i\right)$$

Jeśli składniki d_i są wszystkie dodatnie (ujemne), to $|\delta| \leq \max_i |E_i| \leq n \cdot 2^{-t}$

PRZYKŁAD 1.12

Dane są dwa n -wymiarowe wektory rzeczywiste a i b o składowych odpowiednio $a_i = rd(a_i)$ i $b_i = rd(b_i)$, $i = 1, 2, \dots, n$. Chcemy numerycznie rozstrzygnąć, czy wektory a i b są do siebie prostopadłe. W arytmetyce klasycznej rozwiązanie jest jednoznaczne, sprawdzamy po prostu, czy iloczyn skalarny $S = \sum_{i=1}^n a_i b_i$ równa się zero. Natomiast w rachunku numerycznym nawet dla $S = 0$ możemy otrzymać przybliżenie $\tilde{S} \neq 0$. I odwrotnie, błędy wytworzone mogą spowodować, że dla małych $S \neq 0$ obliczone $\tilde{S} = 0$. Czego więc możemy być pewni? Pozostawiamy do sprawdzenia, że jeśli

$$|S| > 2^{-t} \left(\sum_{i=1}^n m_i |a_i b_i| + |S| \right)$$

to wektory a i b nie są ortogonalne. W przeciwnym przypadku możemy tylko powiedzieć, że w sensie stosowanej arytmetyki wektory a i b są ortogonalne. Nie potrafimy jednak rozstrzygnąć, czy są one rzeczywiście prostopadłe. Chcielibyśmy podkreślić dwojaki charakter rozwiązania numerycznego. Odpowiedź negatywna jest ścisła, a pozytywna – tylko przybliżona. $\square$

Informacja o obliczaniu funkcji standardowych 1.7

W większości maszyn cyfrowych wartości funkcji standardowych $f(x)$ takich jak $\sin(x)$, e^x , $\sqrt{x}$ itp. są obliczane algorytmami numerycznie poprawnymi, tzn.

$$fl(f(x)) = f(x(1+\varepsilon))(1+\delta) \quad (1.31)$$

gdzie $|\varepsilon| \leq K_1 \cdot 2^{-t}$, $|\delta| \leq K_2 \cdot 2^{-t}$, a stałe K_1, K_2 są rzędu jedności. Zainteresowanych metodami obliczania funkcji standardowych odsyłamy do podręczników [9] i [27]. Chcielibyśmy natomiast zbadać, jaki jest błąd względny obliczonej wartości funkcji. Korzystając ze wzoru Taylora dostajemy ze wzoru (1.31) dla $f(x) \neq 0$

$$\begin{aligned} f(x(1+\varepsilon))(1+\delta) &= (f(x) + f'(\xi)\varepsilon x)(1+\delta) \\ &= f(x) \left(1 + \frac{f'(\xi)}{f(x)}\varepsilon x + \delta \right) \end{aligned}$$

gdzie

$$|\xi - x| \leq K_1 \cdot 2^{-t}|x|$$

Błąd względny obliczonej wartości szacuje się zatem przez

$$2^{-t} \left(K_1 \left| \frac{xf'(\xi)}{f(x)} \right| + K_2 \right)$$

Należy pamiętać, że na ogół wartości funkcji standardowych mogą być wyznaczane dla argumentów x ze znacznie mniejszego zbioru niż zbiór liczb reprezentowalnych w danej arytmetyce. Przykładowo, w języku Fortran na maszynie Riad R-32 wartości funkcji $\sin(x)$ mogą być obliczane tylko dla $|x| < \pi \cdot 2^{18}$

Koszt obliczenia wartości funkcji standardowych zależy od konkretnej maszyny cyfrowej i waha się od kosztu kilku do kilkudziesięciu działań arytmetycznych.

Złożoność obliczeniowa

1.8

Dla wielu zadań znane są różne metody i algorytmy ich rozwiązywania. Co roku pojawia się też sporo nowych propozycji. Jest rzeczą naturalną, że spośród tych różnych algorytmów chcielibyśmy wybrać najlepszy. Powstaje więc problem, jakich kryteriów wyboru należy używać. Oczywiście jednym z nich powinna być jakość otrzymanych rozwiązań. Drugim równie ważnym czynnikiem jest koszt danej metody. Jeśli jednak będziemy się ograniczali tylko do porównywania kosztu różnych znanych metod, to nie rozwiążemy w pełni problemu. Znajomość najlepszej metody spośród znanych nie przeczy istnieniu jeszcze lepszej metody.

Dopiero stosunkowo niedawno zaczęto poszukiwać i badać metody optymalne (w sensie przyjętego kryterium) dla danego zadania. Jeśli przyjmiemy za kryterium liczbę działań arytmetycznych potrzebnych do rozwiązania zadania, to metodą optymalną będzie metoda minimalizująca liczbę działań.

Problemami istnienia i własności metod optymalnych oraz ich konstrukcji zajmuje się jeden z działów analizy numerycznej zwany *złożonością obliczeniową* (ang. computational complexity) [3], [15]. Postawienie tego zagadnienia pozwoliło stwierdzić, że wiele klasycznych metod nie spełnia postulatu optymalności. Będziemy wracać do tego w dalszych rozdziałach. Mimo dosyć gwałtownego w ostatnich latach rozwoju złożoności obliczeniowej, tylko dla niewielu, i to stosunkowo prostych zadań, znamy metody optymalne.

Przypomnijmy, że rozważaliśmy dotychczas zadania polegające na obliczeniu dla danych $d = (d_1, d_2, \dots, d_n)$ wyniku $w = (w_1, w_2, \dots, w_m)$

$$w = \varphi(d) \quad (1.31)$$

gdzie

$$\varphi: D \subset R_d \rightarrow R_w$$

Zastanówmy się, jaka jest minimalna liczba działań arytmetycznych $z(\varphi, D)$ potrzebnych do obliczenia wyniku w dla dowolnych danych $d \in D$. Wielkość $z(\varphi, D)$ definiujemy jako

$$z(\varphi, D) = \sup_{d \in D} z(\varphi, d) \quad (1.32)$$

gdzie $z(\varphi, d)$ oznacza minimalną liczbę działań dla obliczenia $\varphi(d)$. Wartość $z(\varphi, D)$ nazywamy *złożonością obliczeniową zadania* (1.31). Rozwiązanie (1.32) jest często bardzo trudne i w sposób istotny zależy od własności zadania. Przyjmując dosyć naturalne ograniczenia dla algorytmów i zadań można wyprowadzić proste oszacowanie z dołu wartości $z(\varphi, D)$.

Zakładamy, że jest znany algorytm rozwiązywania zadania (1.31) dający wynik po wykonaniu skończonej liczby działań. Jeśli algorytm obejmuje również działania logiczne, to ograniczamy zbiór D tak, aby wyniki tych działań nie zależały od d . Będziemy mówili, że rozważane zadanie na zbiorze D ma n danych istotnych, jeśli istnieją dane $d = (d_1, d_2, \dots, d_n) \in D$, dla których zmiana dowolnej ze składowych d_j ($j = 1, 2, \dots, n$) powoduje zmianę wyniku. Ścisłej, jeśli

$$\bigvee_{d \in D} \bigwedge_{1 \leq j \leq n} \bigvee_{\delta} d + \delta e_j \in D \quad \text{ i } \quad \varphi(d) \neq \varphi(d + \delta e_j)$$

gdzie

$$e_j = (0, \dots, 0, 1, 0, \dots, 0)^T$$

Dowodzi się (patrz praca [39]), że jeśli zadanie (1.31) ma n danych istotnych, to minimalna liczba działań $z(\varphi, D) \geq n/2$. Oznacza to, że liczba istotnych danych określa oszacowanie z dołu złożoności obliczeniowej. Oszacowanie to jest często realistyczne, gdyż dla wielu zadań o n istotnych danych są znane algorytmy wymagające wykonania cn działań arytmetycznych, gdzie c jest stałą rzędu jedności. Z drugiej strony istnieją zadania, dla których najtańsze ze znanych algorytmów rozwiązują je kosztem znacznie większej liczby działań niż liczba danych. Problem złożoności obliczeniowej tych zadań jest ciągle otwarty. Przykładowo, dla zadania obliczenia iloczynu dwóch macierzy kwadratowych $n \times n$ minimalna liczba działań $z(n) \geq n^2$. Jednak dopiero w 1969 r. Strassen [73] podał algorytm wymagający mniej niż n^3 działań. Dokładniej, pokazał on, wykorzystując algorytm mnożenia macierzy 2×2 tylko z siedmioma mnożeniami, że $z(n) \leq O(n^{\log_2 7}) \approx O(n^{2.81})$. Co ciekawsze, algorytm ten można stosować także przy rozwiązywaniu układów równań liniowych $n \times n$, znowu kosztem $O(n^{\log_2 7})$ działań. Chcielibyśmy podkreślić, że podana tu charakterystyka kosztu jest charakterystyką asymptotyczną. Dopiero dla bardzo dużych n metoda Strassena jest efektywniejsza od innych

znanych algorytmów wymagających $O(n^3)$ działań. Warto też wspomnieć, że dla wielu zadań można udowodnić (por. praca [54]), iż minimalna liczba działań w algorytmie numerycznie poprawnym ich rozwiązywania musi być istotnie większa od liczby danych.

Problematyka złożoności obliczeniowej jest rozważana także dla zadań, których rozwiązanie wymaga stosowania metod nieskończonych. Oceną efektywności takich metod zajmijmy się przy omawianiu metod rozwiązywania równań nieliniowych.

Uwagi

1.9

Przed przystąpieniem do omawiania metod rozwiązywania konkretnych zadań chcielibyśmy przypomnieć czynniki, które naszym zdaniem należy brać pod uwagę przy numerycznym rozwiązywaniu jakiegokolwiek problemu.

– *Dane.*

Jakie wielkości są danymi rozwiązywanego zadania? Jakie przestrzenie danych i wyników (ich struktury, normy) najlepiej odpowiadają sensowi fizycznemu lub technicznemu rozważanego problemu?

– *Uwarunkowanie zadania.*

Czy zadanie nie jest zbyt wrażliwe na zaburzenia danych? Czy w danej arytmetyce możemy je w ogóle rozwiązać? A może istnieje zadanie równoważne rozważanemu, ale lepiej uwarunkowane? Jeśli odpowiedzi na oba ostatnie pytania są negatywne, to musimy stosować silniejszą arytmetykę. Wiąże się to z użyciem podwójnej precyzji lub nawet z koniecznością korzystania z innej maszyny.

– *Jakość numeryczna stosowanego algorytmu.*

Czy jest on numerycznie stabilny, numerycznie poprawny? Jak duża może być utrata dokładności? Przykładowo, jeśli poziom błędu reprezentacji danych wynosi 10^{-8} , a poziom błędu ich pomiaru $\delta = 10^{-3}$, to użycie algorytmu o wskaźniku stabilności $K \leq 10^5$ nie spowoduje istotnej straty dokładności. Także odwrotnie, znajomość δ i K pozwala na rozstrzygnięcie, czy możemy rozwiązać zadanie z żadaną dokładnością, a jeśli tak, to w jakiej arytmetyce należy przeprowadzać obliczenia.

– *Efektywność stosowanej metody.*

Co wiemy o złożoności obliczeniowej naszego zadania? Jaka jest efektywność innych metod rozwiązujących dany problem? Czy rozpatrywana metoda jest optymalna, tzn. czy miara jej kosztu jest równa złożoności obliczeniowej zadania? Jeśli nie, to czy jest to najtańsza ze znanych metod?

Oczywiście odpowiedź na wiele z tych pytań jest często bardzo trudna.